EMC SCALEIO BASIC ARCHITECTURE DOCUMENTATION

ABSTRACT

This white paper describes the concepts, basic architecture and components of a ScaleIO system, as well as the performance features and analysis of ScaleIO.

March 2017

TABLE OF CONTENTS

EXECUTIVE SUMMARY

This document describes the concepts, basic architecture and components of a ScaleIO system. It is also designed to help users understand the performance features, concepts, and analysis of ScaleIO.

EMC ScaleIO® is software that creates a server-based SAN from local application server storage (local or network storage devices). ScaleIO delivers flexible, scalable performance and capacity on demand. ScaleIO integrates storage and compute resources, scaling to hundreds of servers (also called nodes). As an alternative to traditional SAN infrastructures, ScaleIO combines hard disk drives (HDD), solid state disk (SSD), Peripheral Component Interconnect Express (PCIe) flash cards and NVMe drives to create a virtual pool of block storage with varying performance tiers.

As opposed to traditional Fibre Channel SANs, ScaleIO has no requirement for a Fibre Channel fabric between the servers and the storage. This further reduces the cost and complexity of the solution. In addition, ScaleIO is hardware-agnostic and supports both physical and virtual application servers. It creates a Software-Defined Storage (SDS) environment that allows users to exploit the unused local storage capacity in any server. ScaleIO provides a scalable, high performance, fault tolerant distributed shared storage system.

AUDIENCE

This white paper is intended for employees, partners and customers who have an interest in understanding the concepts of a ScaleIO system.

Terminology

Readers should have general knowledge of the terms listed in this table.

Term	Definition
HDD	Hard disk drives. Traditional magnetic devices that store digitally encoded data.
SSD	Solid state disk that has no moving parts and uses flash memory to store data persistently.
PCIe	Peripheral Component Interconnect Express is a high-speed serial computer expansion bus.
SDS	Within ScaleIO, the SDS is defined as the ScaleIO Data Server. The SDS is the software component that contributes local storage space to an aggregated pool of storage within the ScaleIO virtual SAN. SDS is also an acronym for Software Defined Storage.
SDC	ScaleIO Data Client. A lightweight device driver that exposes ScaleIO shared block volumes to applications.
MDM	ScaleIO Meta Data Manager. Manages, configures and monitors the ScaleIO system.
LAN	Local Area Network providing interconnectivity within a limited (local) area.
OpenStack	Is an open source cloud computing software platform.
CTQ	Command Tag Queueing: reordering the IO to optimize the drive seek and improve the IOPs of the drive
RTO	Recovery time objective. The time for a system/application to be restored, e.g. from back-up, after a failure or disruption.

Term	Definition
DRAM	Dynamic RAM. This is a type of random access memory used in servers for caching, etc.
Protection Domain	A logical container for SDSs. Each SDS can belong to only one Protection Domain.
Storage Pool	A logical cross-SDS group of drives within a single Protection Domain. Usually used to map drives that share common characteristics.
IOC	Basic IO controller. Sometimes called an “HBA”, passes through the IO without any additional features or function.
ROC	RAID-on-Chip. Adds additional features such as buffering Writes, RAID calculations, etc.
DU/DL	Data unavailable/data loss.
DAS	Direct attached storage is local drives in a server.
NAS	Network attached storage is file level storage over a network.
NVMe	Non-Volatile Memory Express, which is a solid state disk with a PCIe interface.

Table 1: Terminology table

ScaleIO System Architecture Basics

ScaleIO systems contain a number of elements including the SDC, SDS and MDM and are discussed in detail in the following sections.

ScaleIO Data Client – SDC

The SDC is a lightweight block device driver that exposes ScaleIO shared block volumes to applications. The SDC runs on the same server as the application. This enables the application to issue an IO request and the SDC fulfills it regardless of where the particular blocks physically reside. The SDC communicates with other nodes (beyond its own local server) over TCP/IP-based protocol, so it is fully routable.

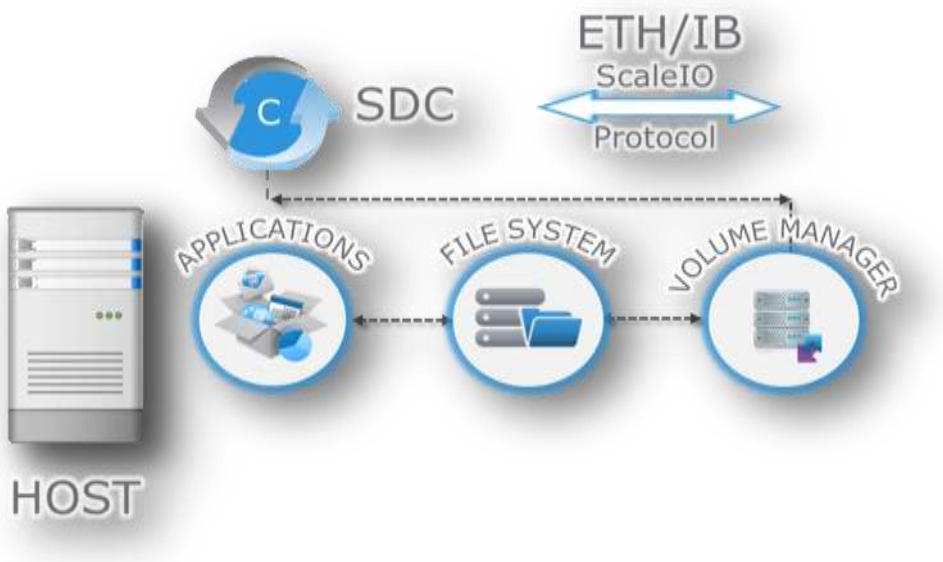

Figure 1 - ScaleIO SDC

Users may modify the default ScaleIO configuration parameter to allow two SDCs to access the same data. This feature provides supportability of applications like Oracle RAC.

ScaleIO Data Server – SDS

The SDS owns local storage that contributes to the ScaleIO Storage Pools. An instance of the SDS runs on every server that contributes some or all of its local storage space (HDDs, SSDs, PCIe, NVMe and flash cards) to the aggregated pool of storage within the ScaleIO virtual SAN. Local storage may be disks, disk partitions, even files. The role of the SDS is to actually perform the Back-End IO operations as requested by an SDC.

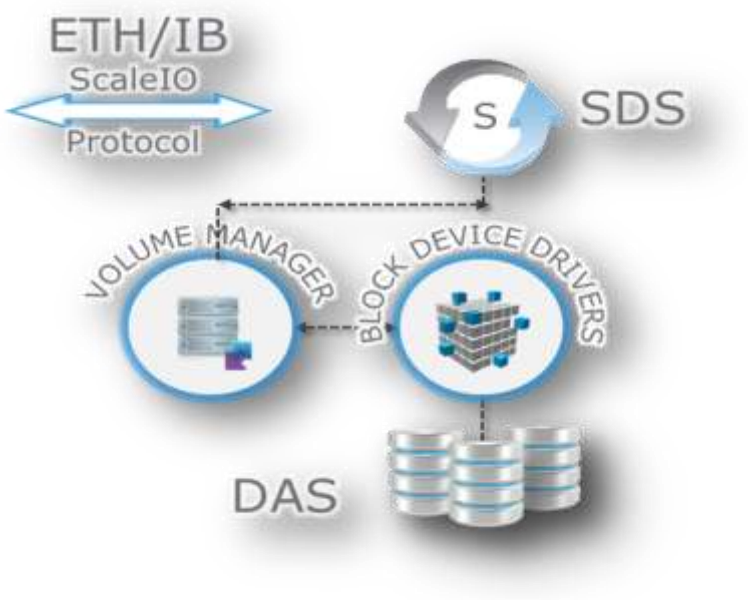

Figure 2 - ScaleIO Data Server

ScaleIO Configurations

There are three standard configurations for ScaleIO implementations, all providing flexibility and scalability. They are as follows and are discussed in the following section.

- Converged Infrastructure (CI) or hyper converged (HCI) configuration
- two-layer configuration
- mixed configuration

Converged Configuration or Converged Infrastructure (CI)

A *converged configuration* involves the installation of both the SDC and the SDS on the same server. This is considered the best practice configuration as well. In this configuration, the applications and storage share the same compute resources and in many ways the storage is like another application running on the server.

The applications perform IO operations via the local SDC. All servers contribute some or all of their local storage to the ScaleIO system via their local SDS. Components communicate over the Local Area Network (LAN).

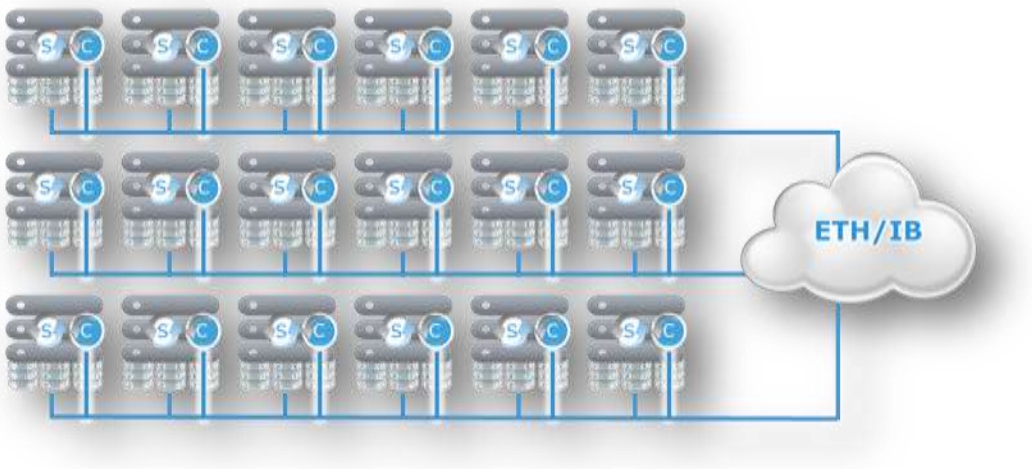

Figure 3 - Converged Configuration

Two-layer Configuration

There is no ScaleIO requirement to implement a Converged configuration, as shown above, where the SDCs and SDSs reside on the same servers.

In certain situations, customers prefer to have the SDS separated from an SDC and installed on a different server. This type of configuration is called a *two-layer configuration* where the SDCs are configured on one group of servers, and the SDSs are configured on another distinct group of servers, as shown in Figure 4.

The applications that run on the first group of servers issue the IO requests to their local SDC. The second group, running SDSs, contributes the servers' local storage in the virtual SAN. Both groups communicate over a local area network. Applications run in one layer, while storage resides in another layer. This deployment is similar to a traditional external storage system such as VNX/Unity, XIO (XtremeIO) and VMAX, but without the Fibre Channel layer.

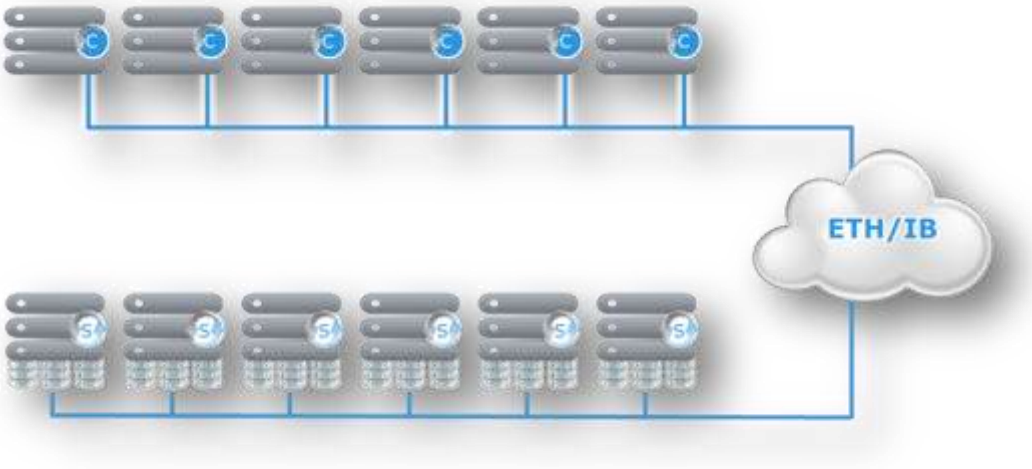

Figure 4 - Two-layer Configuration

Mixed Configuration

ScaleIO is very flexible and allows any combination of the two configurations. When a two-layer and converged configuration exist, this is called a mixed configuration. ScaleIO has no restriction on when configuration changes can be made. This configuration is common in a transient case when moving from a two-layered configuration to a converged configuration.

When a new group of servers are added as SDS servers, ScaleIO will automatically rearrange, optimize and rebalance the data in the background without any downtime. ScaleIO deployments can be changed, or grown quickly - with ease and supporting hundreds of nodes.

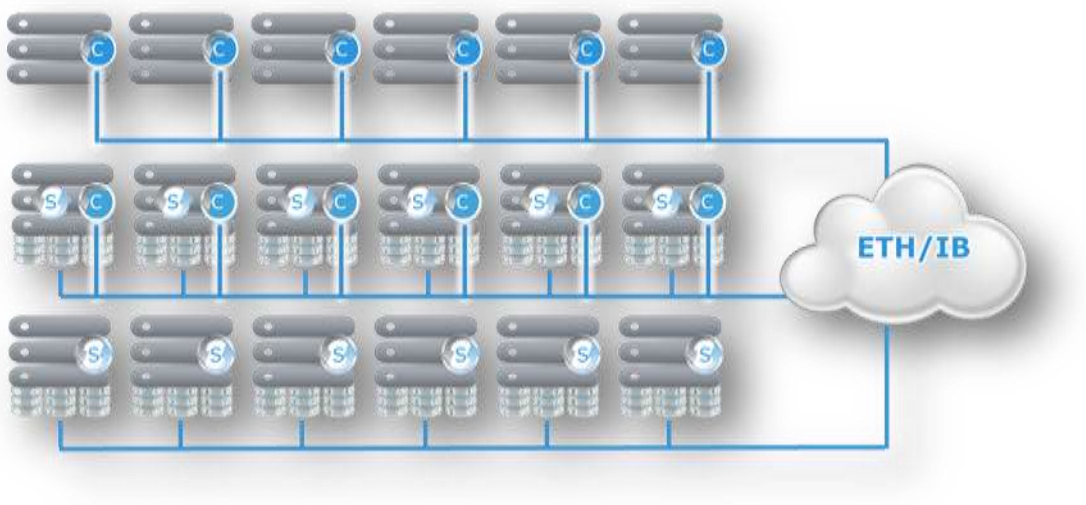

Figure 5 - Mixed configuration consisting of Two-layer and Converged.

Meta Data Manager – MDM

The Meta Data Manager manages the ScaleIO system. The MDM contains all the metadata required for system operation; such as configuration changes. The MDM also allows monitoring capabilities to assist users with most system management tasks.

The MDM manages the meta data, SDC, SDS, devices mapping, volumes, snapshots, system capacity including device allocations and/or release of capacity, RAID protection, errors and failures, and system rebuild tasks including rebalancing. In addition, all user interaction with the system is handled by the MDM. In a normal IO flow, the MDM is not part of the data path and user data *does not* pass through the MDM. Therefore, the MDM is never a performance bottleneck for IO operations.

Currently, an MDM can manage up to 1024 servers. When several MDMs are present, an SDC may be managed by several MDMs, whereas, an SDS can only belong to one MDM. ScaleIO version 2.0 and later supports five MDMs (with a minimum of three) where we define a Master, Slave and Tie-breaker MDM.

The MDM is extremely lightweight and has an asynchronous (or lazy) interaction with the SDCs and SDSs. The MDM daemon produces a heartbeat where updates are performed every few seconds. If the MDM does not detect the heartbeat from an SDS it will initiate a rebuild.

All ScaleIO commands are asynchronous with one exception. For consistency reasons, the *unmap* command is synchronous where the user must wait for the completion before continuing.

Each SDC holds mapping information that is light-weight and efficient so it can be stored in real memory. For every 8 PB of storage, the SDC requires roughly 2 MB RAM. Mapping information may change without the client being notified, this is the nature of a lazy or loosely-coupled approach.

Distributed data layout scheme

ScaleIO's distributed data layout scheme is designed to maximize protection and optimize performance. A single volume is divided into 1 MB chunks. These chunks will be distributed (striped) on physical disks throughout the cluster, in a balanced and random manner. Each chunk has a total of two copies for redundancy and each copy will reside on two different nodes. A third copy is added only when certain operations warrant it; for example, in Instant Maintenance Mode (IMM), if a node goes down, all new writes will be added to a 3rd temporary copy

It is important to understand that the ScaleIO volume chunks are not the same as data blocks. The IO operations are performed at the block level. If an application writes out 4 KB of data, only 4 KB are written, not 1 MB. The same goes for read operations—only the required data is read.

Note: Two copies are never stored on the same physical server. They are "meshed" throughout the cluster.

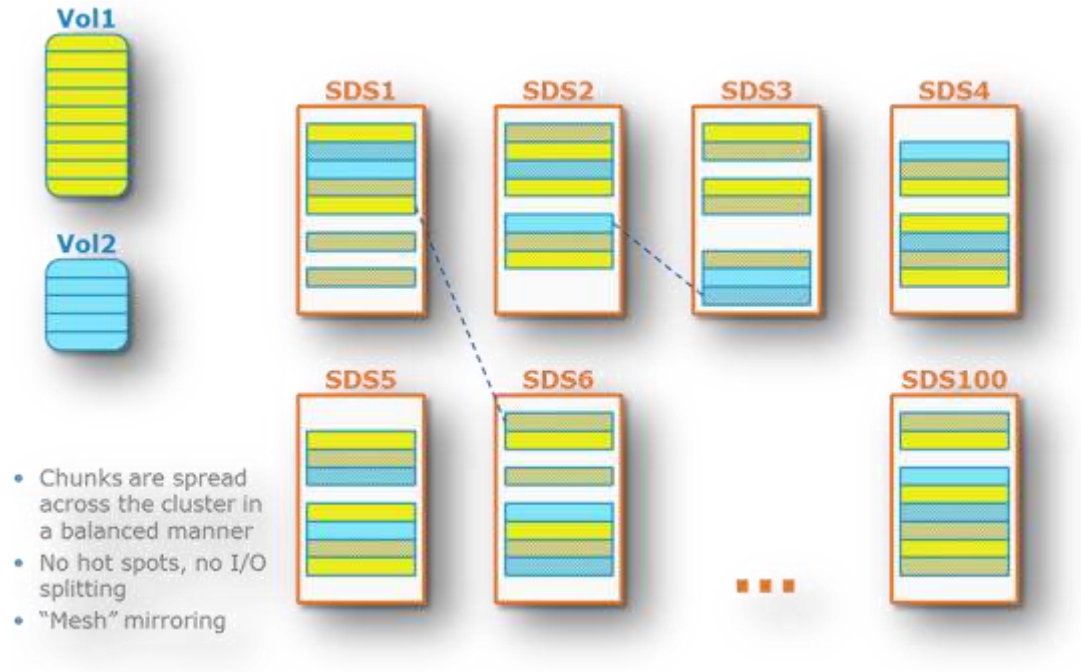

Figure 6 - ScaleIO volume layout

Rebuilds

ScaleIO systems automatically rebuild a failed drive or failed server. For example if SDS1 crashes, ScaleIO will rebuild its 1 MB chunks by copying from its mirrors. This process is called a *forward rebuild*. It is a many-to-many copy operation, which is what makes the rebuild such a quick operation.

Upon completion of the forward rebuild operation, the system is fully protected and optimized. Better still, while this operation is in progress, all of the data is accessible to its applications so that users experience ***no*** outage or disruption in service.

The *backward rebuild* option is used when a node goes down for only a short period of time. In ScaleIO 2.0 and later, the size of each chunk during a backward rebuild is 4KB (and not 1MB as in previous versions). This option is managed by the MDM that determines whether updating the mirrored volumes will be faster than rebuilding the data on the downed server. The MDM collects all tracked changes from all SDSs, and is therefore equipped to make the best decision on forward or backward rebuild methods.

ScaleIO always reserves space on servers in the case of an unplanned outage, when rebuilds are going to require unused disk space. To ensure data protection during server failures, ScaleIO reserves 10% of the capacity by default (other percentages are allowed based on the rule in the next paragraph), not allowing it to be used for volume allocation.

To ensure full system protection during the event of a node failure, users must ensure that the spare capacity is at least equal to the amount of capacity in the node containing the maximum capacity, or the maximum Fault Set capacity (the term Fault Set is described later in the doc). If all nodes contain equal capacity, it is recommended to set the spare capacity value to at least 1/N of the total capacity (where N is the number of SDS nodes).

Rebalance

One of ScaleIO's greatest benefits is its elasticity (flexibility). Adding or removing devices (drives), and/or servers to a ScaleIO configuration triggers an automatic migration and rebalance of the remaining devices.

During this process, the data is migrated and then rebalanced across the servers. The rebalance process simplifies ScaleIO management, eliminating long refresh cycles, and making the environment more dynamic and flexible. If more compute power or storage is required, simply add more devices or more servers.

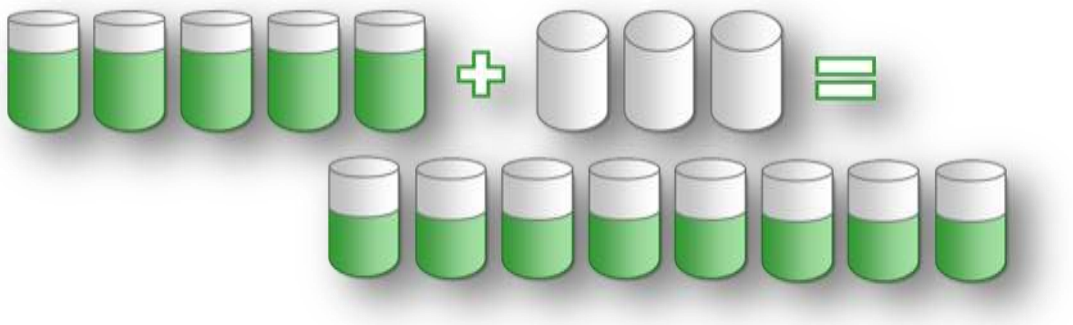

Figure 7 - Rebalance after adding 3 devices or nodes

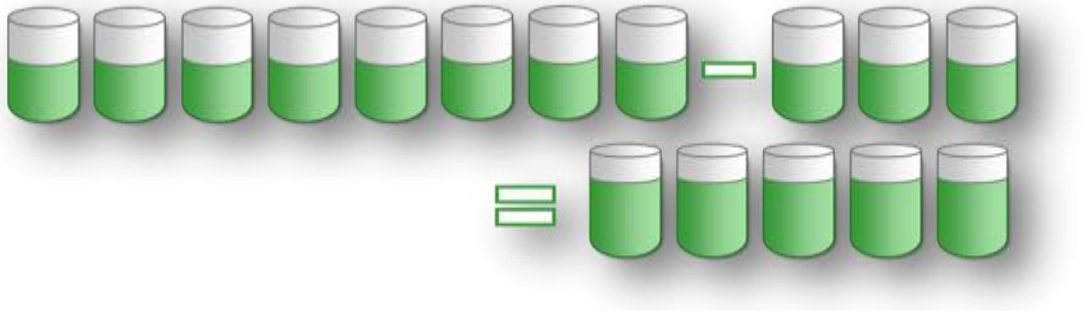

Figure 8 - Rebalance after removing 3 devices or nodes

Note: The aggressiveness of the rebuild/rebalance operation can be controlled by the user. By default, rebuilds are set to be a higher priority than rebalance because of the availability aspects of rebuild. Refer to the Rebalance/Rebuild throttling section on page 18 for more details.

IO Flow

IOs from the application are serviced by the SDC that runs on the same server as the application. The SDC fulfills the IO request regardless of where any particular block physically resides.

When the IO is a Write, the SDC sends the IO to the SDS where the Primary copy is located. The Primary SDS will send the IO to its local drive and in parallel, the IO is sent to the secondary mirror located on a secondary SDS. Only after an acknowledgment is received from the secondary SDS, the primary SDS will acknowledge the IO to the SDC.

A Read IO from the application will trigger the SDC to issue the IO to the SDS with the Primary chunk.

In terms of resources consumed, one host Write IO will generate two IOs over both the network and back-end drives. A read will generate one network IO and one back-end IO to the drives. For example, if the application is issuing an 8 KB Write, the network and drives will get 2x8 KB IOs. For an 8 KB Read, there will be only one 8 KB IO on the network and drives.

Note: The IO flow does not require any MDM or any other central management point. For this reason, ScaleIO is able to scale linearly in terms of performance.

Every SDC knows how to direct an IO operation to the destination SDS. There is no flooding or broadcasting. This is extremely efficient parallelism that eliminates single points of failure. Since there is no central point of routing, all of this happens in a distributed manner. The SDC has all the intelligence needed to route every request, preventing unnecessary network traffic and redundant SDS resource usage.

Read Cache

Cache is a critical aspect of storage performance. ScaleIO uses server DRAM for Read RAM Cache (RMcache) as well as SSD/Flash devices (RFcache) for caching reads. ScaleIO cache uses recently-accessed (LRU) data readily available to manage caching. IOs read from cache have a lower response time than IOs serviced by the drives.

Another benefit of caching IO is that it reduces the data drive workload which in many cases is a performance bottleneck in the system.

Cache in ScaleIO is managed by the SDS. It is a simple and clean implementation that does not require cache coherency management, which would have been required if cache were managed by the SDC.

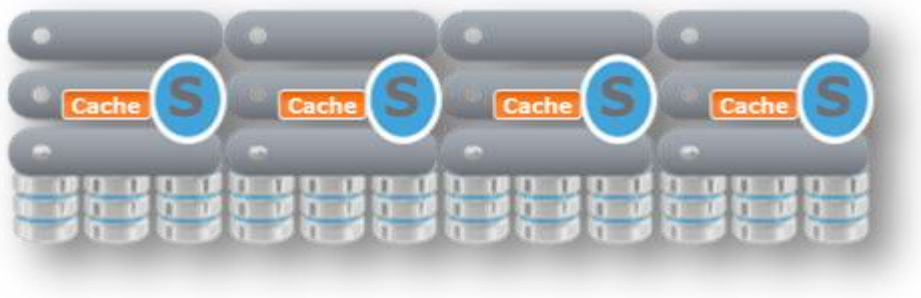

Figure 9 – Cache managed by the SDS

Read RAM Cache

Read RAM Cache, or RMcache, is an SDS feature that uses server DRAM memory for read caching.

Read cache characteristics:

- No caching of rebuild and rebalance IOs (because there is no chance these IOs will be reused)
- Writes are buffered for Read after Write IOs
    - Default = ON
- Unaligned writes are not buffered
- The cache page size is 4 KB
    - If IO is smaller than 4 KB or not aligned to 4 KB, the back-end IO will be aligned to 4 KB. This is a type of pre-fetch mechanism.
- Cache size (per SDS)
    - Default = 128 MB
    - Cache can be resized dynamically and disabled on the fly.
    - Max Cache Size = 300 GB
- Cache can be defined for each Storage Pool
- Cache can be defined per Volume
- Max IO size cached: 128 KB

Cache is managed by two main data structures:

- User Data (UD): This is a copy of the disk data residing in Cache
    - pre-allocated in one continuous buffer
    - divided into 128 KB blocks
    - managed using an efficient Least Recently Used (LRU) algorithm
- Meta Data (MD): Contains pointers to addresses in the UD
    - MD uses Hash with two inputs (keys): physical LBA, Device number

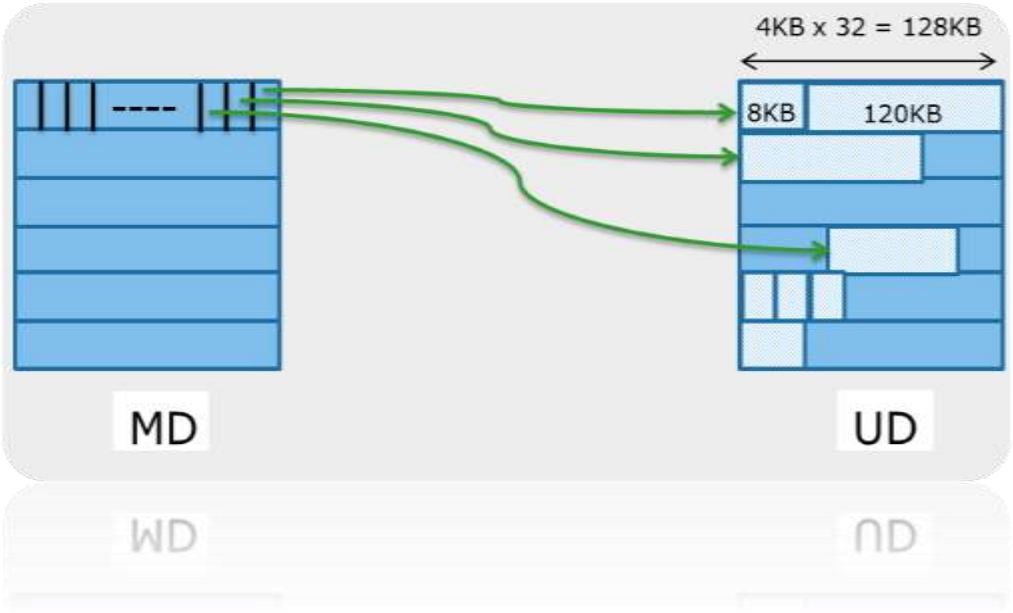

Figure 10 - Cache Meta data pointers and User Data structures

Note: Both the MD and UD use Least Recently Used (LRU) algorithms to make sure "old" data is evicted from cache first.

Read RAM Cache (RMcache) can help increase ScaleIO performance. If the node is storage only (in other words; the node is only used for ScaleIO) then the recommendation is to turn on Read RAM Cache for HDD pools and use as much of the server DRAM as possible.

In a converged configuration (where ScaleIO is sharing the server with other applications), depending on the available DRAM resources, it may also help to turn on Read RAM Cache for HDD pools and increase the cache size from the default (128MB) to approximately 2 GB depending on the available memory.

It is not recommended to use RMcache for SSD pools. Although there is a small latency benefit, the extra overhead of managing the cache which can significantly benefit HDD pools, may reduce the max performance available from the very fast SSD pools.

Read Flash Cache

Read Flash Cache (RFcache) uses SSD/Flash devices in the server as read cache devices. This feature is used to increase read performance and buffers writes to increase the performance of Read-after-Write IOs. It allows users to create and configure an "RFcache" device for any SSD or Flash card. ScaleIO will cache user data depending on the mode selected. The RFcache device is also referred to as an accelerated device.

The benefit of RFcache over RMCache is the larger size available with SSD/Flash devices vs. memory, so for a lower cost the user can gain a much larger cache which can greatly improve performance for some workloads.

As always with cache devices, it takes time for the cache to warm up and reach its maximum potential. This is especially an issue with tests and POC where it takes time to capture the benefit of Read Cache – see Figure 11.

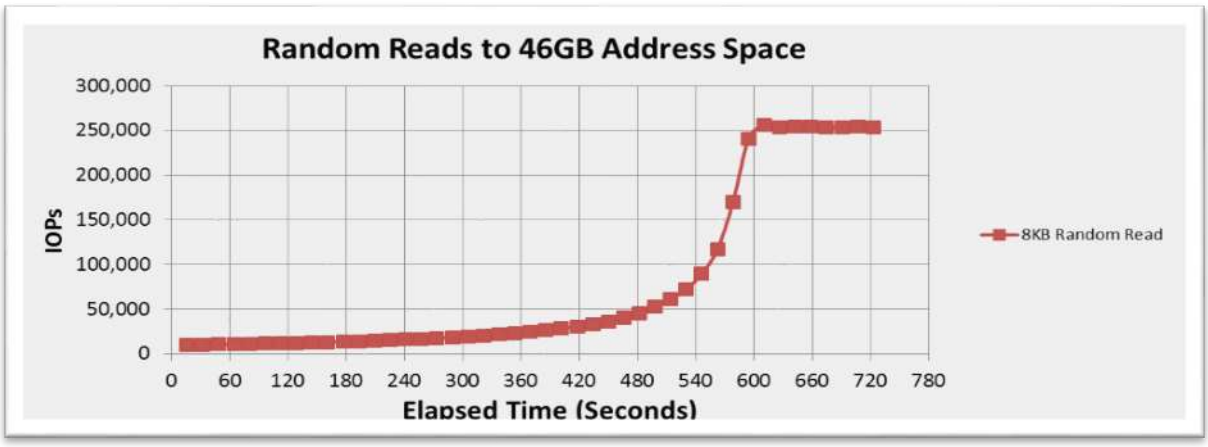

Figure 11 - Cache warming up

It is not recommended to use RFCache for SSD pools because there is no benefit in using SSD to cache the reads for SSD devices.

Write buffering

Write buffering has many advantages. One benefit is a reduction in Write Response Time which is much lower when the IO is acknowledged from a write buffering DRAM/Flash device, rather than a HDD drive.

An added benefit of buffering writes is that it utilizes the "elevator reordering". This is sometimes referred to as Command Tag Queuing (CTQ). Elevator reordering minimizes the seek on the drive and increases the max IOPs of an HDD drive, and even reduces the drive's load because of rewrites to the same address locations.

Note: Apart from the re-write effect and CTQ, write buffering does not affect sustained random writes maximum throughputs.

Write buffering can be achieved in various ways. With RMCache and RFCache, Writes are only buffered for Read after Write caching. Another way to achieve Write buffering is to use Raid controllers (e.g. LSI/Broadcom, PMC etc.) that have write buffering. It is important that the DRAM buffer be protected against sudden power outages to avoid any data loss.

Other write buffering options include using SSD/Flash drives configured in the systems. This allows increasing cache to the size of the SSD device (from 1-4 GB of Raid controller DRAM).

A newer option for write buffering is introduced with NVDIMMs. This option allows protecting the server memory in case of a power cycle. This means that applications can write to memory and assume that memory will be volatile in case of a power cycle.

Sizing of Write cache is not always a simple task. With Read Cache, users almost always get a benefit from having more cache, and the challenge is more on the cost vs. IOP gain. A good rule of thumb would be 1-10% of the useable capacity. With write cache, having a very large cache does not always provide benefits. This is why Raid controllers do a fine job even with a small DRAM cache (1-4GB). However, an under-sized cache will not be good enough to buffer a large write burst. A properly sized write cache is mostly a function of the write burst the server is expected to have. For example, many applications flush the server memory every few seconds. In such cases, the write cache size should be similar to the size of the server DRAM to accommodate for the burst of writes. Sizing cache much larger will have some benefit in write coalescing, but usually that benefit does not grow with the size, so adding more write cache is good, but up to a limit.

For Flash only configurations, it is usually recommended to use a pass-through HBA instead of the Raid controller, or use NVMe drives that are connected directly to the PCI bus. The reason is similar to the Read cache of SSD/Flash pools. Although the latency might be slightly lower, the overhead of managing the write buffering will add overhead that reduces the max IOPs of the flash pool, therefore, it's better to let the IOs go directly to the drives.

IO Types

There are three types of IO operations in a ScaleIO system:

- Read Hit
- Read Miss
- Write

Each IO type and size behaves differently since they exercise different components inside the ScaleIO system.

Read Hits (RH)

A *Read Hit* is a read to the ScaleIO system (SDS) where it finds the requested data already in the Read Cache space. Therefore, RHs run at memory of flash speeds, not HDD disk speeds, and there are no HDD disk operations required.

Read Misses (RM)

A *Read Miss* is a read to the ScaleIO system when requested data is not in cache and must be retrieved from physical disks (HDD or SSD). Reads that are serviced by the Raid controller's cache are still considered a Read Miss from ScaleIO's management point of view. It's important to consider that sequential reads are not counted separately. If any IO is serviced from the host read cache, those IOs are counted as Read Hits. Any other IO is counted as Read Misses.

Note: There is minimal pre-fetch as part of the ScaleIO cache code. For example, a sequential read of 512B will bring 4 KB into cache. A best practice recommendation is to use the Read-ahead feature in the Cache controller only for HDD drives. This allows pre-fetching IOs to increase the performance when using HDD drives. Read-ahead buffers are used to improve sequential reads by reading the consecutive address spaces into DRAM before the host read arrives, making the host reads from DRAM and not disk. With Flash drives, this feature is not necessary and not recommended.

Writes

A *Write* is a Write IO operation to the ScaleIO system. Apart from the write buffering cases described in the previous section, there is little difference between the various write types, e.g. Sequential and Random.

Protection Domains

A Protection Domain is a set of SDSs. Each SDS belongs to one (and only one) Protection Domain. Thus, by definition, each Protection Domain is a unique set of SDSs.

The ScaleIO Data Client (SDC) is not part of the Protection Domain. An SDC can reside on the same server as an SDS that belongs to Protection Domain X can also access data in Protection Domain Y.

Using a Protection Domain has the following benefits:

- Reduces the impact of simultaneous multiple failures in large clusters
- Performance isolation when needed
- Data location control (e.g., multi tenancy)
- Helps to fit network constraints

The recommended number of nodes in a protection domain as well as the number of drives in a storage pool, should be set according to sizing best practices to ensure availability of the cluster. The simplest way to achieve this is to use the ScaleIO sizing tool.[1] Users

[1] The ScaleIO sizing tool can be found at https://scaleio-sizer.emc.com/

can add Protection Domains during installation. In addition, Protection Domains can be modified, post-installation, with all the management clients.

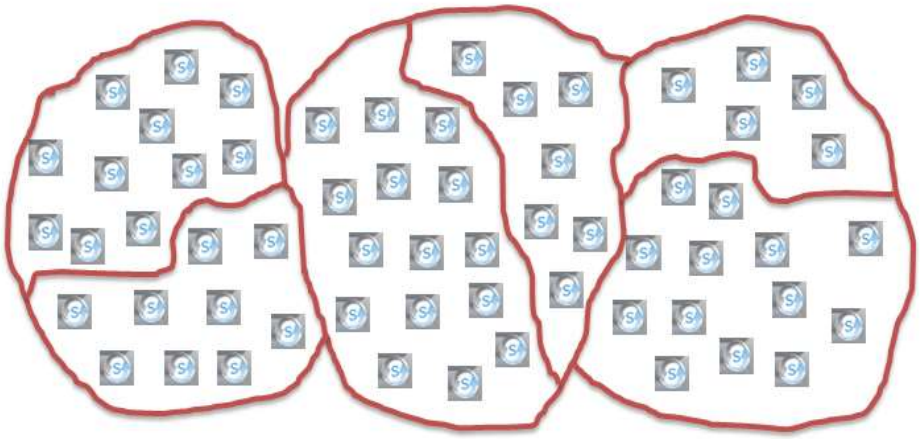

Figure 12 - Protection Domains – protecting a set of ScaleIO Data Servers

Storage Pools

Storage Pools allow the generation of different storage tiers in the ScaleIO system. A Storage Pool is a set of physical storage devices in a Protection Domain. Each storage device (drive) belongs to one (and only one) Storage Pool. When a Protection Domain is generated, it has one Storage Pool by default. Storage Pools are mostly used to group drives based on drive types and drive speeds, e.g. SSD and HDD.

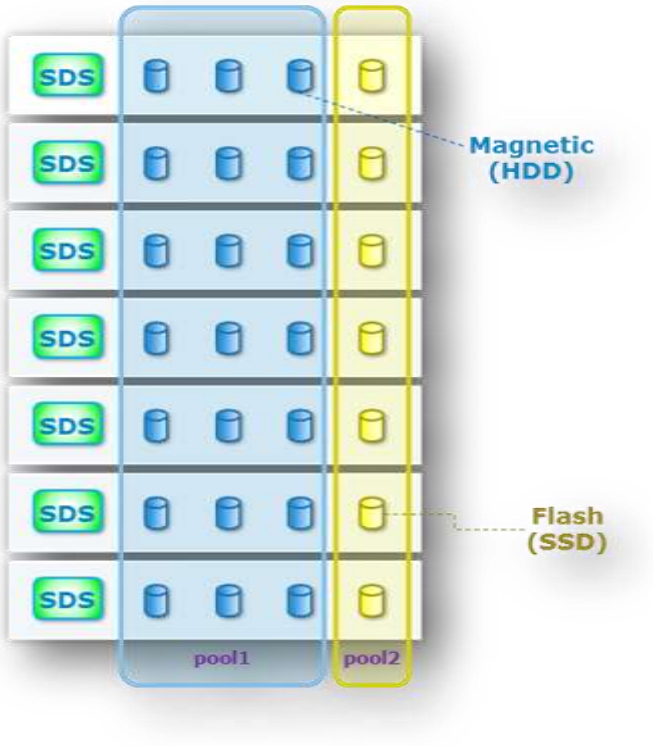

Figure 13 - Different Storage Pools within a Protection Domain

Fault Set

In many cases, data centers are designed such that a unit of failure may consist of more than a single node. An example use case is where a rack contains several SDSs and the customer wants to protect the environment from a situation where the whole rack fails, or is lost by a rack power outage or some disaster.

The fault set will limit mirrored chunks from being in the same fault set. A minimum of 3 fault sets is required per protection domain. Deploying Fault Sets will prevent both copies of data from being written to SDS's in the same fault set. This ensures that one copy of the data is available in the event that an entire fault set fails.

It is vital to keep in mind that the MDMs will also need to be spread across the Fault sets.

An important consideration with Fault sets is the spare capacity. When we have 3 fault sets, we need to allow one fault set to fit in the other two in case of a failure. This means that the spare space will need to be 34% (when all fault sets are equal). This can add significant overhead which is why, in many cases, customers use fault sets when they have around 10 and more because then the spare would be 10% which is standard.

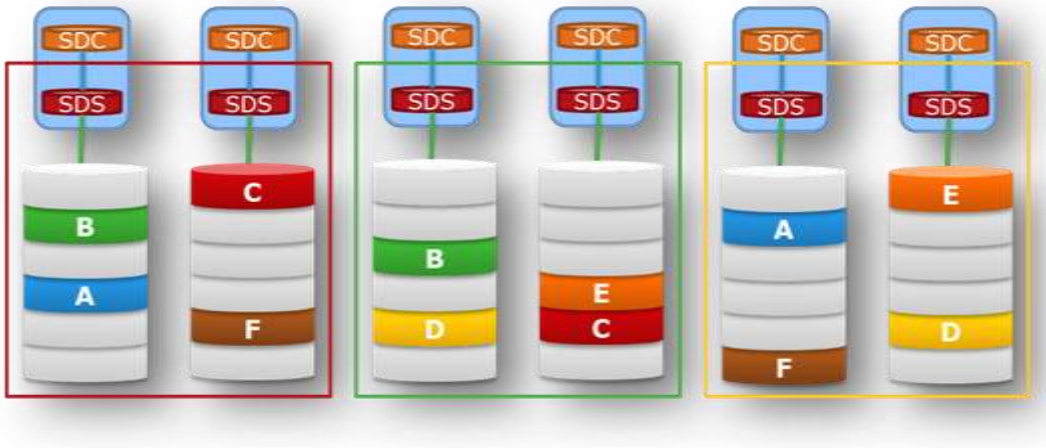

Figure 14 - Fault set data distribution

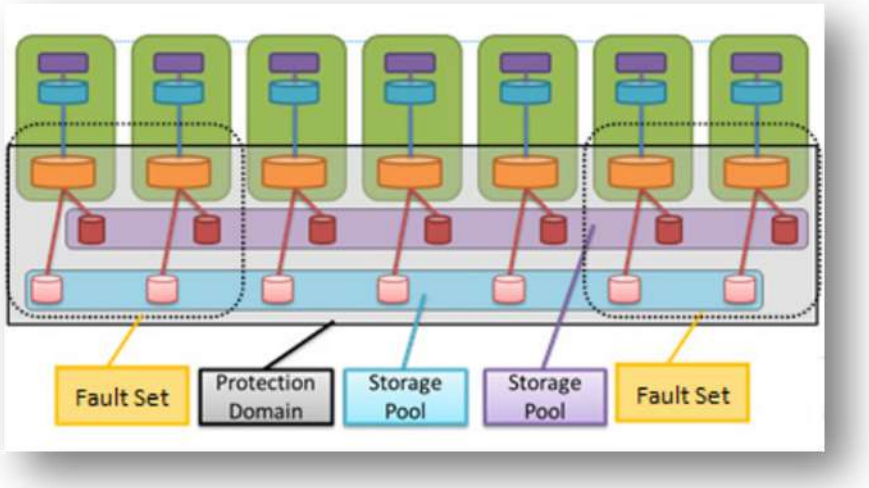

Figure 15 - Example Configuration; Fault Sets, Storage Pools and Protection Domain.

Snapshots

The ScaleIO storage system enables users to take snapshots of existing volumes, up to 31 per volume (future versions will allow 127). All snapshots are thin provisioned. Once a snapshot is generated, it becomes a new, unmapped volume in the system. Users manipulate snapshots in the same manner as any other volume exposed to the ScaleIO storage system.

All snapshots taken together form a consistency group. They are consistent in the sense that they were all taken at the same point in time. If there is a contextual relationship between the data contained by all the snapshot members, then that set is meaningful. The consistency group allows manipulation of the entire set.

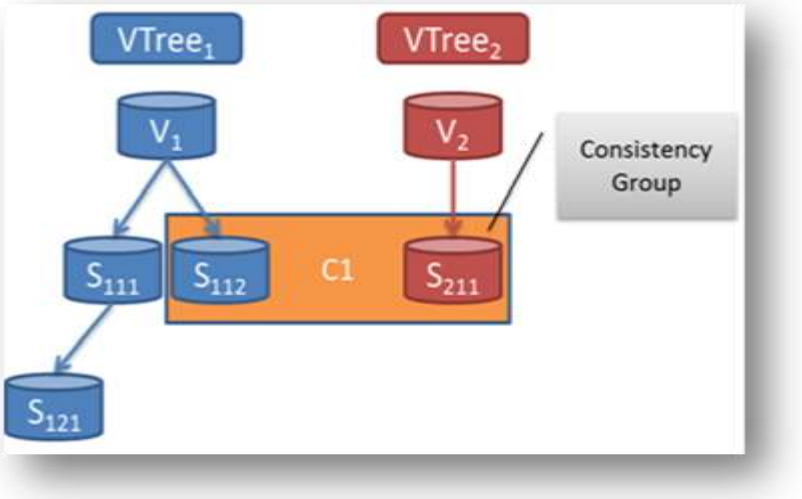

Figure 16- Snapshot operations

This structure related to all the snapshots resulting from one volume is referred to as a VTree (or Volume Tree). It's a tree spanning from the source volume as the root, whose siblings are either snapshots of the volume itself or descendants of it.

Each volume has a construct called a vTree which holds the volume and all snapshots associated with it. The limit on a vTree is 32 members (future versions will allow 128) – so 1 is taken by the original volume and the rest are available for snapshots.

Quality of Service (QoS)

ScaleIO has several forms of QoS options to control the performance in various cases.

One use case is limiting the workload per volume. Users can adjust the amount of IOPs and bandwidth that one SDC can generate for a volume. These parameters are configured using the ScaleIO CLI (Command Line Interface), and the REST interface, on a per-client/per-volume basis.

Another use case is changing the default values of rebuild, rebalance and migration. These operations have an inherent impact on the system because they consume resources from the system (mostly drive resources). The trade off in these cases is between; the impact to the host applications, and time to complete the operation. The defaults are set, such that a rebuild will be completed quickly and rebalance and migration will have a low impact. These defaults can be changed to speed up or slow down these operations.

Rebalance/Rebuild throttling parameters:

Setting these parameters allows users to set the rebalance/rebuild IO priority policy for a Storage Pool. It determines the priority policy that will be imposed to favor application IO over rebalance/rebuild IO.

There are four possible priority policies that may be applied:

- **No Limit**: No limit on rebalance/rebuild IOs. This option will help complete the rebuild/rebalance ASAP, but may have an impact of the host applications
- **Limit Concurrent IO**: Limit rebalance/rebuild number of concurrent IOs per SDS device.
- **Favor Application IO**: Limit rebalance/rebuild in both bandwidth and concurrent IOs.
- **Dynamic Bandwidth Throttling:** Limit rebalance/rebuild bandwidth and concurrent IOs according to device IO thresholds. This option helps to increase the rebalance/rebuild rate when the host application workload is low.

The Response Time Law

Queuing theory is the mathematical study of waiting lines (or queues). Much of computer performance analytics and capacity planning rely on queuing theory. It is an academic discipline in itself, but the part we are concerned with here is the Response Time, governed by Little's Law which states that for every distribution and every queuing policy the Average-Response-Time = Queue length * Service Time.

Queue length = Utilization / (1-Utilization) is true only when the inter arrival distribution is exponential.

Little's Law

The Response Time Law applies to the ScaleIO world in that, as components become busier and busier, they eventually hit the knee of the curve (this is when utilization Response Time begins to skyrocket).

Figure 17 shows an example of the Response Time Law in action. In other words, we need to keep close track of internal system component utilizations if we want ScaleIO systems to have good response times.

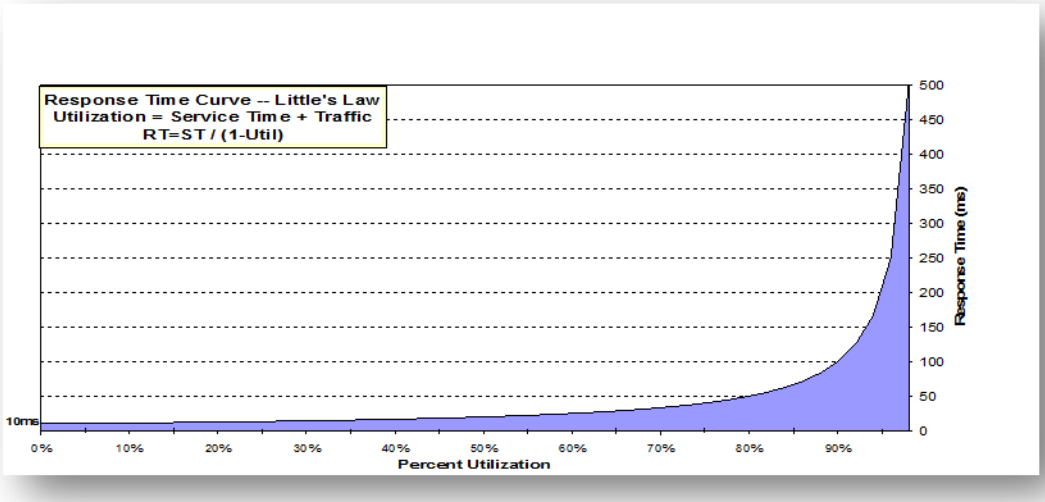

Figure 17- Response time law

An estimate of the queue length is very similar (Assuming service time is random (exponential) and inter-arrival time has the Poisson distribution[2].

From this formula, it is simple to see that at 50% utilization, the queue becomes larger than 1 (one IO being serviced and another IO waiting) and then the Response Time starts to grow. Between 65% and 70% utilization the queue is already at around 2 (one IO being serviced and another two IOs are waiting). In many cases, this increase becomes a concern in terms of Response Time.

Access Density

Access density is the number of IOPs per GB. This is a measurement of performance capability.

On a drive-to-drive comparison, spinning drives (10K RPM, 15K RPM, 7.2K RPM) perform nearly the same. However, on a capacity-to-capacity basis, they won't perform the same. For example:

- 600 GB drives can deliver only about half the performance of 300 GB drives

[2] Poisson distribution expresses the probability of a given number of events occurring in a fixed interval of time and/or space, if these events occur with a known average rate and independently of the time since the last event.

Put another way, a system with 24x10K RPM drives can handle roughly the same number of IOPs, regardless of whether the 24 drives are 300 GB in size, or 400 GB or 600 GB. Since any of these 10K RPM drives will deliver approximately the same IOPs, the access density will vary depending solely on the capacity of the drives.

Maximum Workload Calculations

In most cases, performance documents only publish the maximum IOPs capability of a system for very simple use cases. For example: max 8 KB Writes/sec and max 8 KB Reads/sec. However, real workloads are usually a mix of

Reads and Writes and the question becomes, how do users estimate the maximum performance of mixed workload, from those of basic published parameters.

As an example, let's assume there is a storage system that is capable of doing a maximum of 30,000 8 KB Writes/sec or a maximum of 120,000 8 KB Reads/sec.

Note: These are the real limits of certain Raid Controllers.

The question then becomes; what will be the maximum number of IOPs for this system if the application's host workload is 30% Reads and 70% Writes?

A naïve answer would be: 30,000*0.7 + 120,000*0.3 = 57,000 max IOPs.

However, this assumption is incorrect. A hint: something is wrong if we calculate the Write portion of this workload: 57,000*0.7 = 39,900 Writes/sec. This of course cannot be correct because the max Writes/sec in this system can do (hypothetically) 30,000 IOPs. Therefore, it makes no sense that with extra Reads the systems can do more Writes.

The correct way to calculate the maximum IOPs is to convert the maximum number of IOs per second, into seconds Per IO. In other words, calculate the overhead time per IO, then add up the times, and invert back to IOs per second.

Table 2 demonstrates the calculation.

	Writes/sec	Reads/sec	Total Time	Max IOPs
Max IOPs	30,000	120,000		**38,710**
Time (usec)	33.33	8.33	25.83	
			=Writes*0.7+Reads*0.3	1/(Total Time)

Table 2- Correct way to calculate maximum IOPs

The Table 2 example uses the front end IOPs, but takes into account back end limits.

Clearly, this result is very different from the naïve calculation, which is why this is a very important point to keep in mind.

The best way to perform workload calculations is by using the ScaleIO sizing tool. The tool presents information about the availability of the configurations. ScaleIO clusters can be designed to have enterprise level availability (5-6x9) and can compete with any other Enterprise level storage system.

You will find the ScaleIO sizing tool at;**Error! Hyperlink reference not valid.** https://scaleio-sizer.emc.com/

Reliability and Availability

Every drive has a finite "life time" which is usually measured in terms of MTBF: Mean Time Between Failures. An enterprise level spinning drive (HDD) has about 800,000 hours of MTBF, which equates to around 90 years. This may seem like a lot, however, when you have 90 drives it means you could have a drive failure every year. With 900 drives in a system, you could potentially have a drive fail every ~5 weeks.

Because the drives are protected, one drive failing isn't an issue by itself. The risk of a DU/DL starts after a drive fails. When another drive from the same "Raid protection group" fails, we end up with a DU/DL situation.

It's important to understand the definition of Reliability and Availability.

Reliability: is the probability of not having a failure event during some period of time.
Availability: is the percentage of time the data will be available.

In other words reliability is the absence of failure. Availability relates to duration of time that the system is operating normally.

ScaleIO systems can be designed to have 6'9s which are in line with the best enterprise storage systems. For more details please use the ScaleIO sizer: https://scaleio-sizer.emc.com/

The difference between Marketing "k" and Engineering "K"

In most science use cases, a 'k' = kilo = 10^3 = 1000. However, in computer science and other related areas, a 'K' = 2^{10} = 1024. In practice, some customers and Marketing folks use the 'k' (base-10) definition and some use the 'K' (base-2) definition.

ScaleIO uses the 'K' (base-2) definition! These different definitions lead to different numerical values for the same amount of storage.

For example,

10,000,000,000 bytes = 10.000 GB_{10} = 9.313 GB_2

The varying calculation leads to significantly different definitions for the larger increments of storage:

		GB_{10}	GB_2
Prefix	Capacity	'k' = 1000	'K' = 1024
Kilo-	KB	1000	1024
Mega-	MB	1000*1000	1024*1024
Giga-	GB	1000*1000*1000	1024*1024*1024
Tera-	TB	1000*1000*1000*1000	1024*1024*1024*1024
Peta	PB	1000*1000*1000*1000*1000	1024*1024*1024*1024*1024

Table 3 - 'k=1000' vs. 'K=1024' Capacities

Note: Drive capacities are generally published using the marketing K, while storage capacities are managed using the eng K. This leads to about 10% difference between the published capacity of the drive and the actual capacity as seen by ScaleIO.

ScaleIO Limits

The following table defines ScaleIO's version **2.0** different configuration limits. For best practices, refer to this table before implementing a ScaleIO system.

Item	Limit
ScaleIO System Raw capacity	300 GB – 16 PB
Device Size	100 GB – 8 TB
Minimum Storage Pool capacity	300 GB
Volume Size	8 GB – 1 PB
Maximum number of volumes/snapshots in system	32,768 *
Maximum number of volumes/snapshots in Protection Domain	32,768
Maximum number of volumes + snapshots in single VTree	32
Maximum capacity per SDS	64 TB
SDSs per system	1,024
SDSs per Protection Domain	128 *
Maximum devices (disks) per SDS server	64
Maximum devices (disks) per Storage Pool	300 *
Minimum devices (disks) per Storage Pool	3, on different SDSs
Maximum SDCs per system	1,024
Maximum volumes that can be mapped to a single SDC	8,192
Maximum Protection Domains per system	256
Maximum Storage Pools	1,024
Maximum Storage Pools per Protection Domain	64
Maximum Fault Sets per Protection Domain	64
Max IP addresses per MDM server	4
Maximum IP addresses per server (SDS)	8
RAM Cache	128 MB – 300 GB

- Any implementation demanding higher limits will require an RPQ (request for product qualification).

References

Probability, Statistics, and Queueing Theory, 2nd Edition, Arnold O. Allen, Academic Press, 1990.